SQLUG

SCOTTISH QL USERS GROUP

June 1999

NUMBER 110

THE SCOTTISH QL USERS GROUP NEWSLETTER
Estimated SQLUG Accounts 1st April 1996 to 31st March 1999

LAST MEETING AT PORT OF MENTEITH VILLAGE HALL

SUNDAY 9TH MAY

George arrived at our meeting with a brand new tower case containing a Q40, which he proceeded to put to use. We did not get to see the innards but I was given the manual to read, although it was much too technical for me to take in in the time available. To all appearances its behaviour was the same as the computers George has brought to other meetings I certainly was not aware of any cursing from that part o the hall. In a departure from normal, Giles was also investigatinc software. He has purchased the Linux manual which comes with 5 CD Roms and John Sadler was assisting in getting the systen up and running on an Intel machine . This is a free standing operating system so you don't need MSDOS or QDOS already present in the machine. Included in the set are the usual utilities like word processor and spread sheet. As well Giles had his miniQL going but this only pointed up the limitations of our darling's graphics as Linux has a number of screens which showed what Giles's flat screen SVGA monitor could really do when attached to a decent driver. Here I go again '

"C!" GEORGE

John Sadler

C was originally developed to write operating systems easily. Hence it is very powerful and also you can create havoc easily!

MALLOC & CALLOC

When using malloc and calloc it is essential to check that it returns a real address and not null. If it returns null, terminate the program or else you may find your program going anywhere in your system. When freeing memory after allocating it with malloc and calloc it is essential to make the pointer pointing to the address equal to null before freeing the memory else you may access it later with disastrous results. Programs that do not do this have memory leakage problems.

GEORGE'S ARRAY CONUNDRA

To understand them it is essential to understand pointers in C. When you declare a variable in C the variable in actual fact points to the location of memory that holds the variable. So if

you declare a variable "int x = 4;" then &x;is the address of the variable and if it is read in the correct manner will return 4. Also C lets you do the opposite, "int x;" creates a pointer to the location holding the integer data, and this pointer is called x. Furthermore the compiler knows that x points to an integer. Do not forget you must assign a value to it by say "x = 4;" before you access it, as it not initialised when created.

Arrays work in a similar manner. So the declaration "int s[] = (1,2,3);" creates an integer array with three integers in it 1, 2, 3; and "int *s[3];" creates an integer array of size 3 with s pointing to the first integer unknown, putting in *s[l] = 1;" and "*s[2] = 2". Now C knows that this is an array of integers so *s++i points to s + 1 times size of integer which is 2.

So moving to George's conundra the line "int in1[3] = (1, 3, 5>;" creates an array of 3 integers called in1 and "int2[2] = (7, 9);" creates another array in2 of 2 integers. "int in[] = {inl, in2};" creates a pointer to an array of the arrays inl and in2. Hence "*in[0][1]" asks for the contents of the second element of the first array in the array of arrays? which is 3. Also "*in[1]" asks for the contents of the second array, which is the first element, so it returns that which of course is 7. This sort of programming is definitely not recommended. Now "(*in)[l]" may return the second element of the first array by implication which is 3, or else if you are unlucky it returns anything. If you are the lucky, compiler will object with an error message.

"(*++in)[1]" will return an error message because the compiler does not know how to interpret cast (incrementing the contents) of by one, and therefore returns an error message. Cast changes the format one type, say integer, to another type, say floating point or a pointer, in memory. "*++in[1]" increments the pointer to in by one i.e. the second array and get the contents of the second element which is 9.

For the next four items "**" the contents of the contents does not exist and therefore the compiler will return an error message. "*++*in" is intrepreted as *(++(*in)) i.e. get the contents of in which is the first array, increment the pointer by and one get its content which is the second element of the first array of 3.

"++*++*in** increments the result by one and hence returns 4. "in" prints the pointer or address of the array in. For the next four items the compiler does not know what the pointer incremented is and returns an error message fortunately. Finally "in[0]" and "in[1]" prints the two elements of in which are the locations or pointers to inl and in2.

!!Please do not ask me to explain the explanation!!

[Seems sufficiently explicatory to me, but you do need to have a familiarity with the syntax of C and to read this in conjunction with George's article split between March and April SQLUG Ed]

NEXT MEETING

PORT OF MONTEITH VILLAGE HALL, SUNDAY 13th JUNE

11am - 5pm

Our meeting is on the usual second Sunday of the month which is a little late this time. Note this is our usual loaation but next month we are moving to Thornhill five miles to the East on the Stirling Road (A81 then A873. Full details of this next month together with a nap.

CONTACTS

Here is a full list of the membership other than for one member who wishes to be ex-directory.

LIBRARY

The best way to get material from the library is to come to our monthly meeting. George Gwilt has agreed to offer the postal service so if you can't manage along you can always send to George for disks by post. To do so, send the appro- priate number of formatted disks, together with adequate return postage to

[redacted]

Give the reference numbers of the disks from the library index when you are ordering. Remember also that the index disk will need to be updated from time to time, and can be sent back ordering. The standard format is 720 Kb 3 1/2 inch disk, but if your format is different, we still may be able to help you. Contact me in the first instance.

LIBRARY BOOKS

Our LIBRARY BOOK collection is looked after by George, although there are also a large number of books held by different members of the group. We have copies of most books published about the QL, and George has produced a dbf file. If you let him know that you want to borrow a particular book, he will bring it to the next meeting, or tell you who is looking after it.